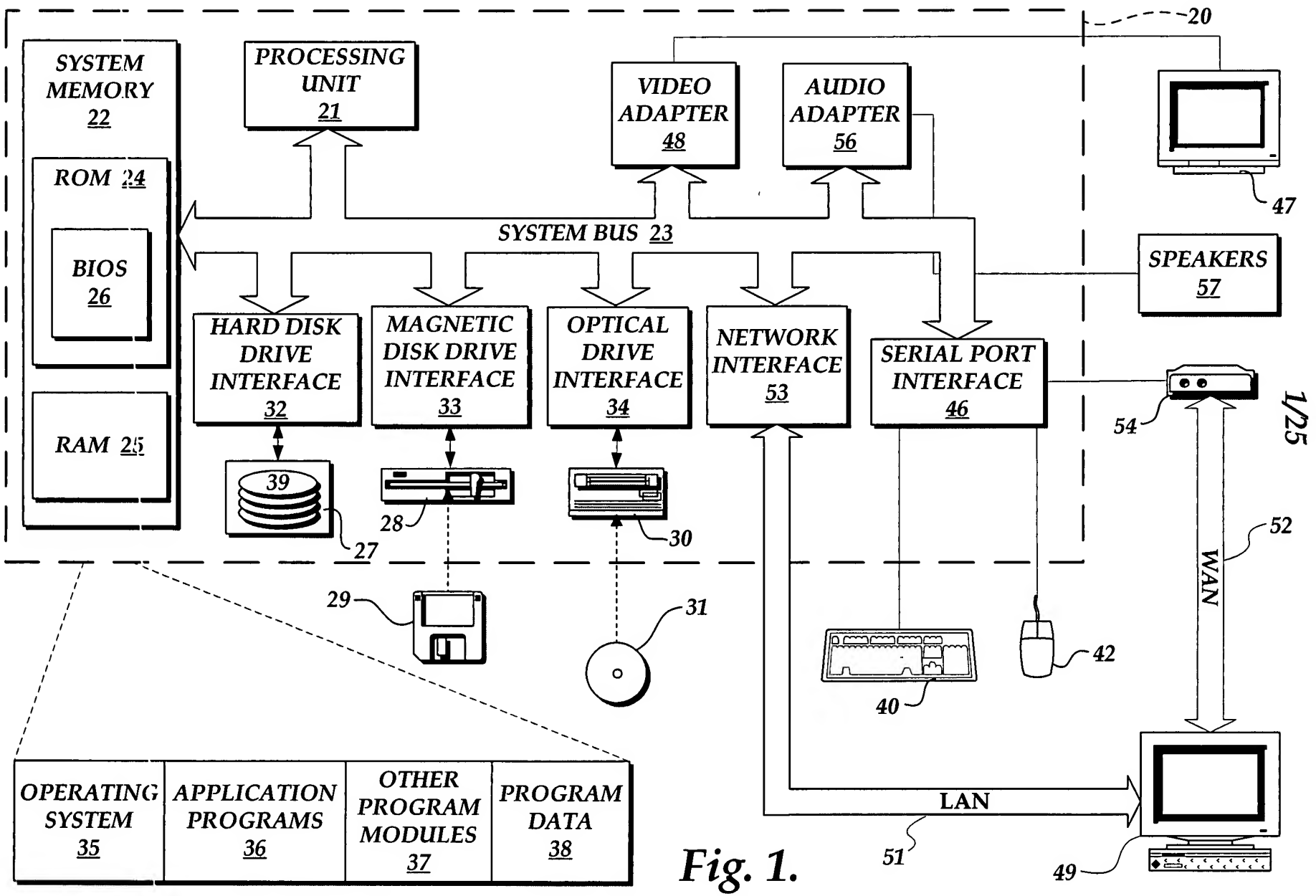
20
SYSTEM MEMORY 22
ROM 24
BIOS 26
RAM 25
PROCESSING UNIT 21
VIDEO ADAPTER 48
AUDIO ADAPTER 56
47
SYSTEM BUS 23
SPEAKERS 57
HARD DISK DRIVE INTERFACE 32
MAGNETIC DISK DRIVE INTERFACE 33
OPTICAL DRIVE INTERFACE 34
NETWORK INTERFACE 53
SERIAL PORT INTERFACE 46
54
52
WAN
39
27
28
30
29
31
42
40
LAN
51
49
OPERATING SYSTEM 35
APPLICATION PROGRAMS 36
OTHER PROGRAM MODULES 37
PROGRAM DATA 38

Fig. 1.

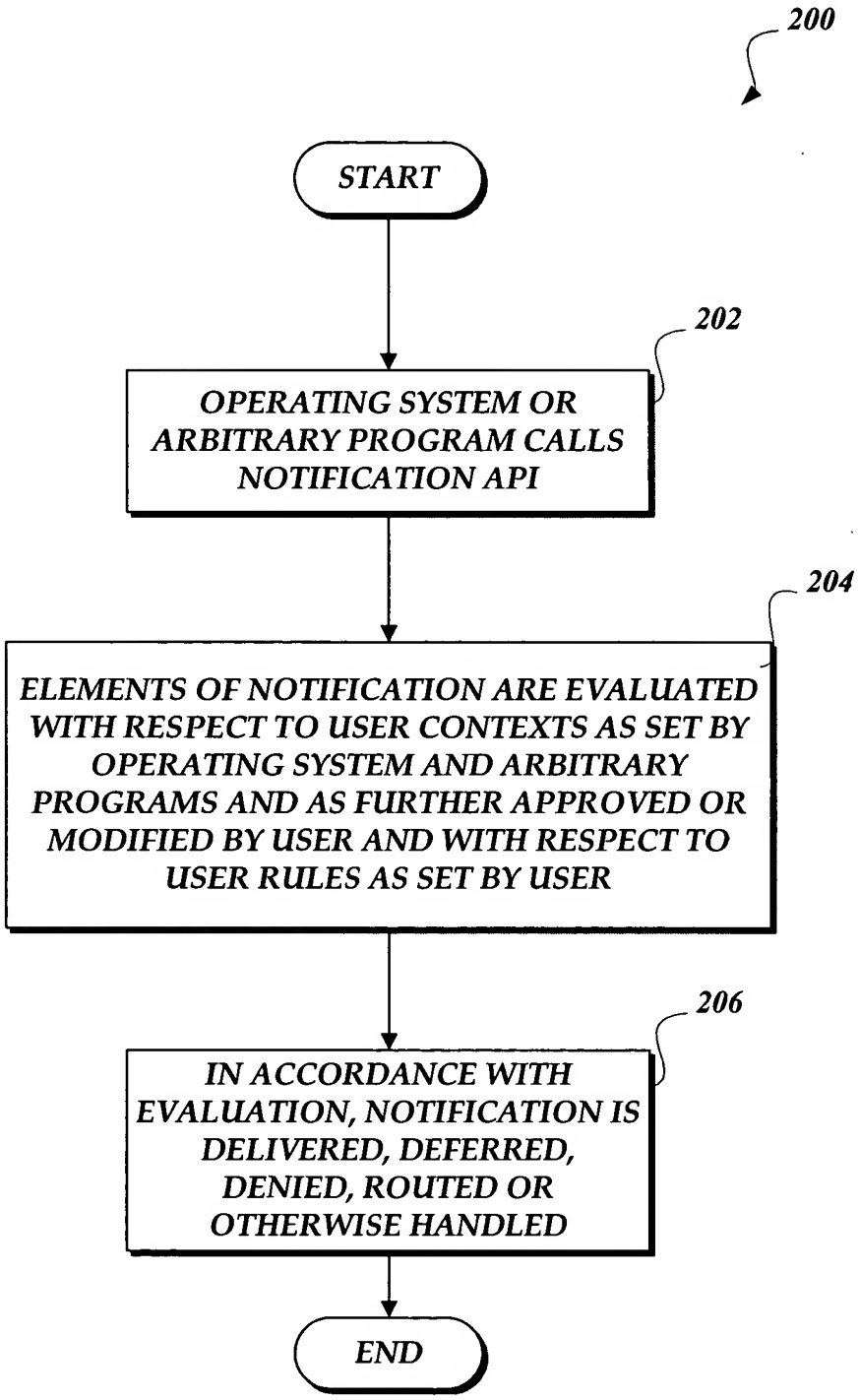
200
START
202
OPERATING SYSTEM OR ARBITRARY PROGRAM CALLS NOTIFICATION API
204
ELEMENTS OF NOTIFICATION ARE EVALUATED WITH RESPECT TO USER CONTEXTS AS SET BY OPERATING SYSTEM AND ARBITRARY PROGRAMS AND AS FURTHER APPROVED OR MODIFIED BY USER AND WITH RESPECT TO USER RULES AS SET BY USER
206
IN ACCORDANCE WITH EVALUATION, NOTIFICATION IS DELIVERED, DEFERRED, DENIED, ROUTED OR OTHERWISE HANDLED
END

Fig.2.

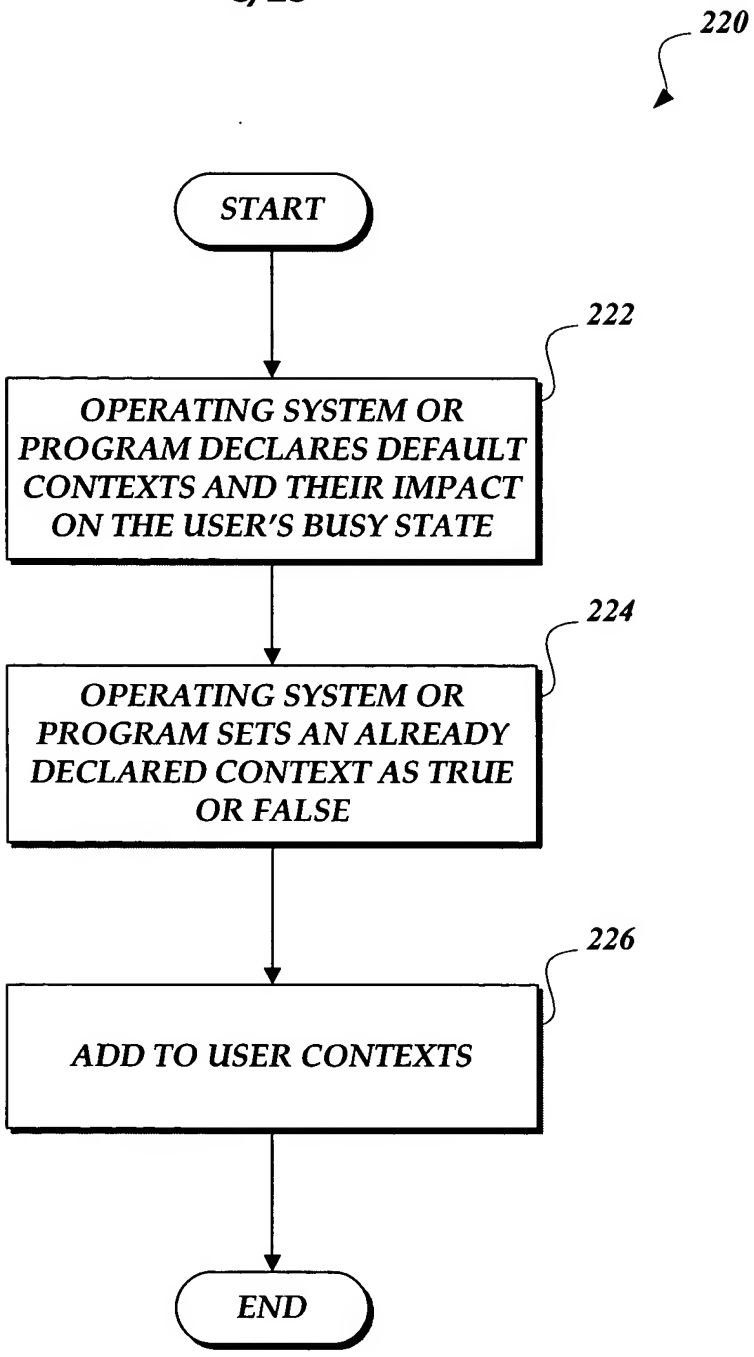
220
START
222
OPERATING SYSTEM OR PROGRAM DECLARES DEFAULT CONTEXTS AND THEIR IMPACT ON THE USER'S BUSY STATE
224
OPERATING SYSTEM OR PROGRAM SETS AN ALREADY DECLARED CONTEXT AS TRUE OR FALSE
226
ADD TO USER CONTEXTS
END

Fig.3.

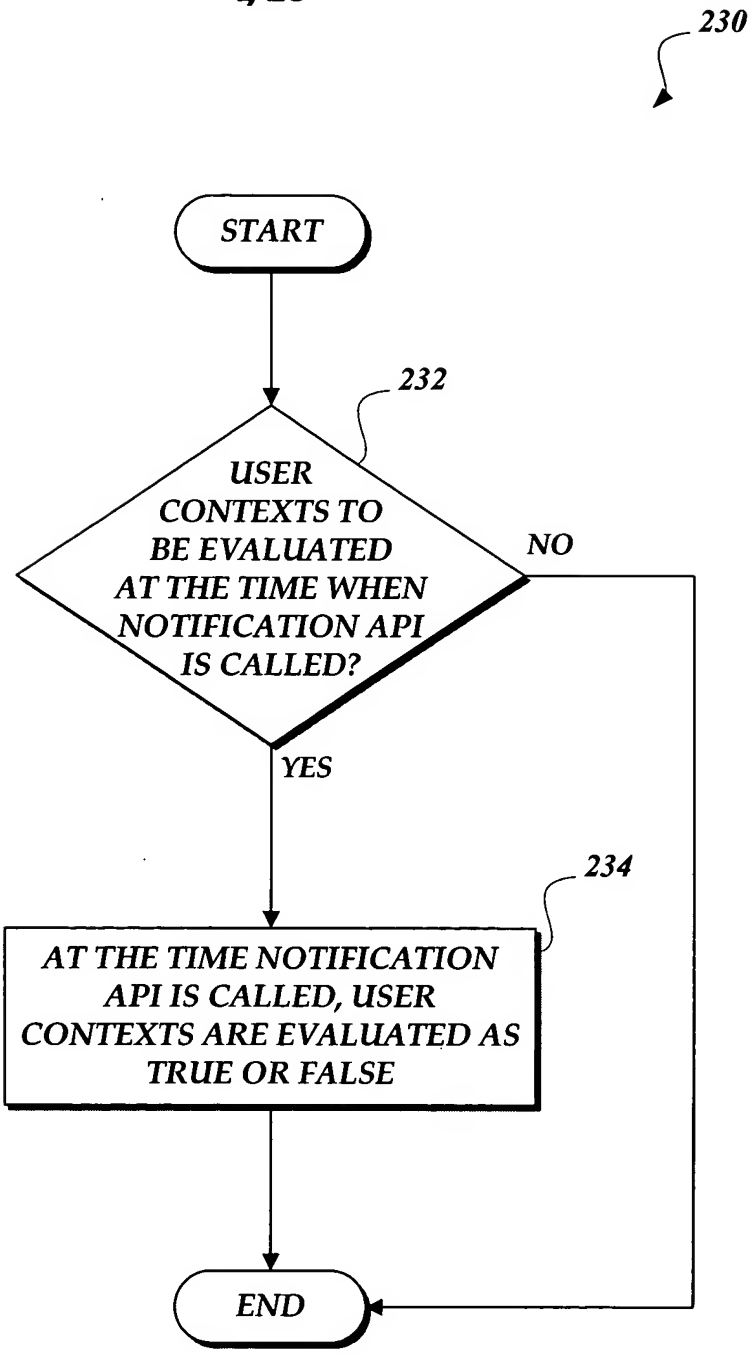
230
START
232
USER CONTEXTS TO BE EVALUATED AT THE TIME WHEN NOTIFICATION API IS CALLED?
NO
YES
234
AT THE TIME NOTIFICATION API IS CALLED, USER CONTEXTS ARE EVALUATED AS TRUE OR FALSE
END

Fig.4.

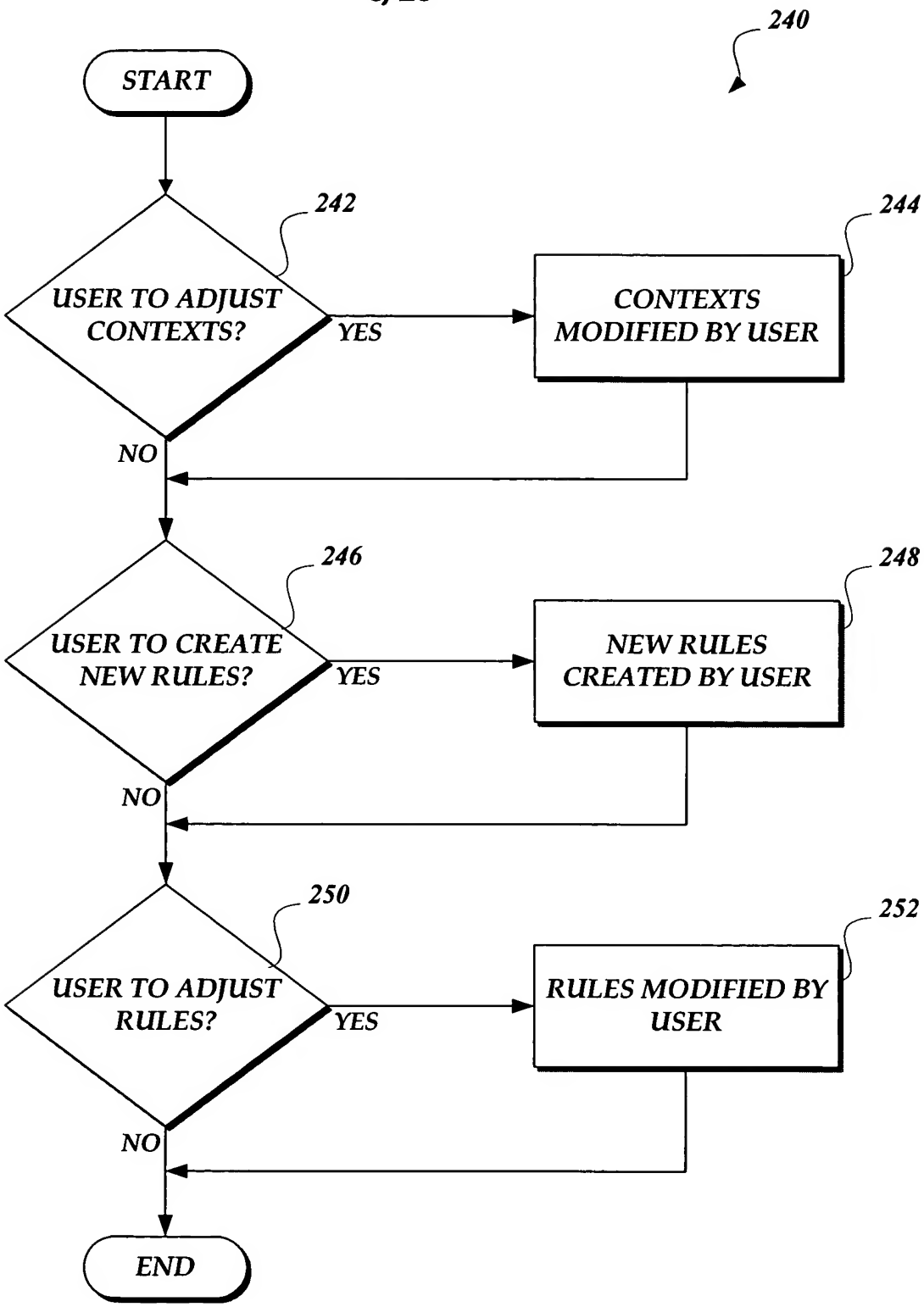
240
START
242
USER TO ADJUST CONTEXTS?
YES
244
CONTEXTS MODIFIED BY USER
NO
246
USER TO CREATE NEW RULES?
YES
248
NEW RULES CREATED BY USER
NO
250
USER TO ADJUST RULES?
YES
252
RULES MODIFIED BY USER
NO
END

Fig.5.

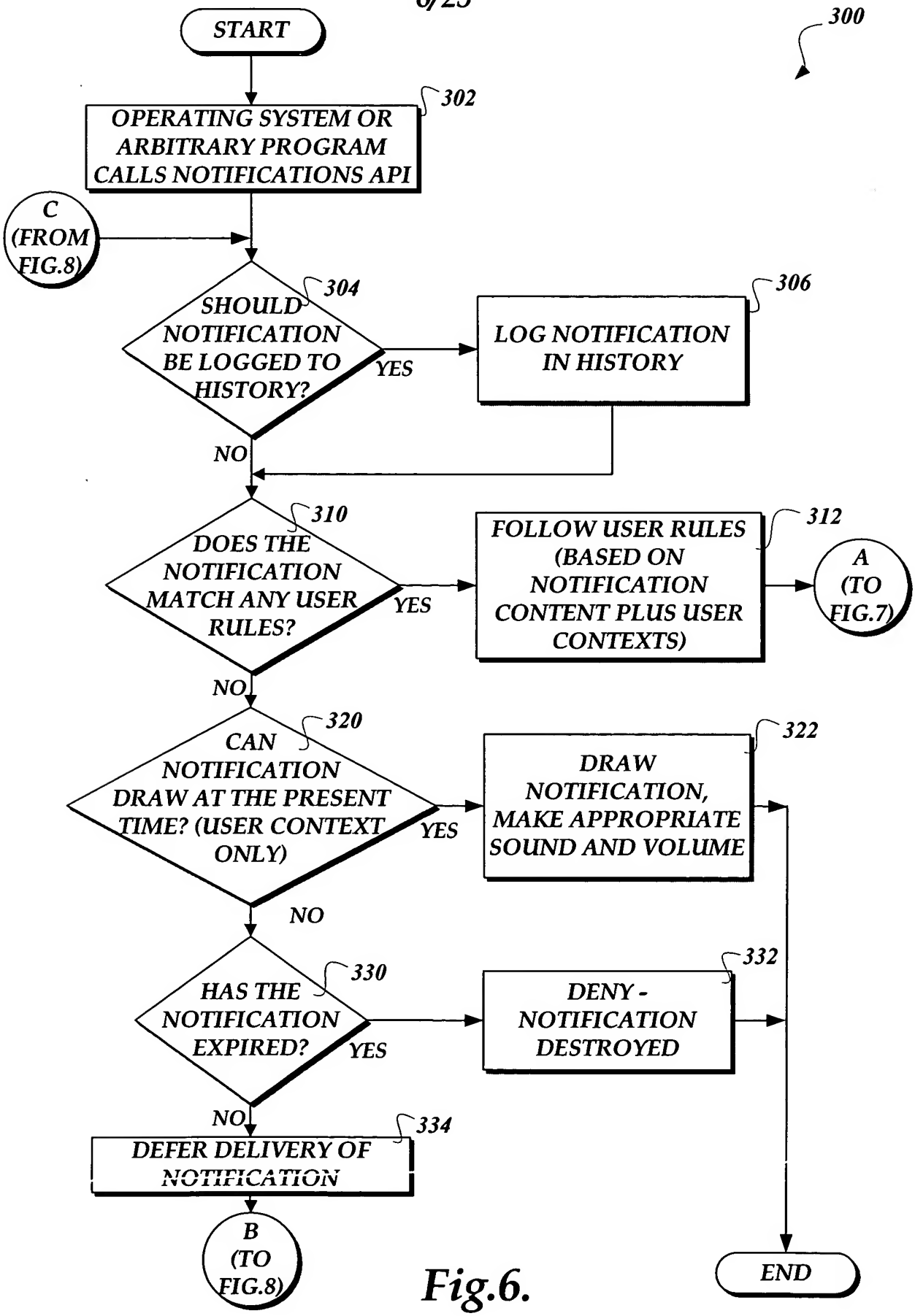
300
START
302
OPERATING SYSTEM OR ARBITRARY PROGRAM CALLS NOTIFICATIONS API
C (FROM FIG.8)
304
SHOULD NOTIFICATION BE LOGGED TO HISTORY?
YES
306
LOG NOTIFICATION IN HISTORY
NO
310
DOES THE NOTIFICATION MATCH ANY USER RULES?
YES
312
FOLLOW USER RULES (BASED ON NOTIFICATION CONTENT PLUS USER CONTEXTS)
A (TO FIG.7)
NO
320
CAN NOTIFICATION DRAW AT THE PRESENT TIME? (USER CONTEXT ONLY)
YES
322
DRAW NOTIFICATION, MAKE APPROPRIATE SOUND AND VOLUME
NO
330
HAS THE NOTIFICATION EXPIRED?
YES
332
DENY - NOTIFICATION DESTROYED
NO
334
DEFER DELIVERY OF NOTIFICATION
B (TO FIG.8)
END

Fig.6.

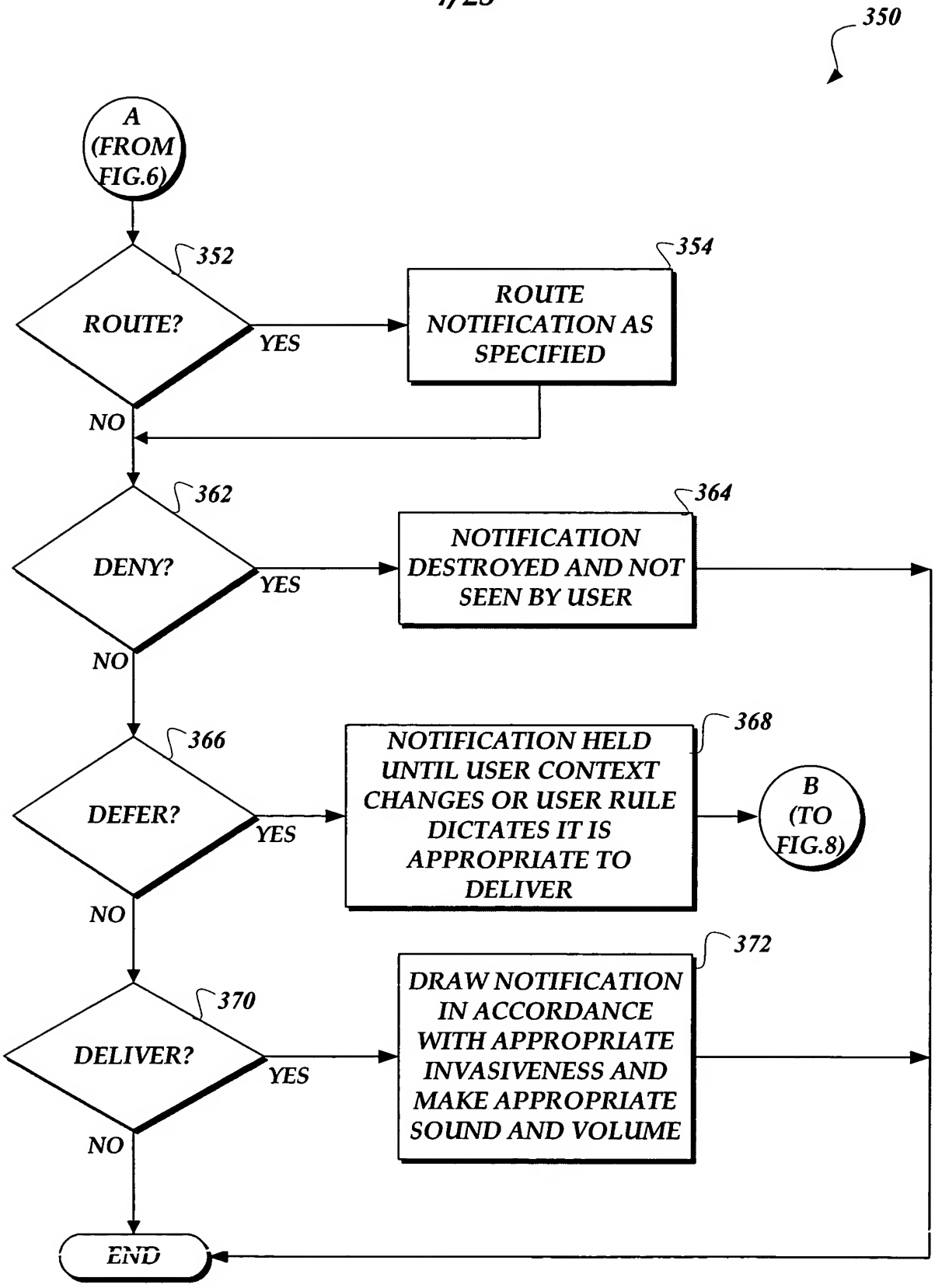
350
A (FROM FIG.6)
352
ROUTE?
YES
354
ROUTE NOTIFICATION AS SPECIFIED
NO
362
DENY?
YES
364
NOTIFICATION DESTROYED AND NOT SEEN BY USER
NO
366
DEFER?
YES
368
NOTIFICATION HELD UNTIL USER CONTEXT CHANGES OR USER RULE DICTATES IT IS APPROPRIATE TO DELIVER
B (TO FIG.8)
NO
370
DELIVER?
YES
372
DRAW NOTIFICATION IN ACCORDANCE WITH APPROPRIATE INVASIVENESS AND MAKE APPROPRIATE SOUND AND VOLUME
NO
END

Fig.7.

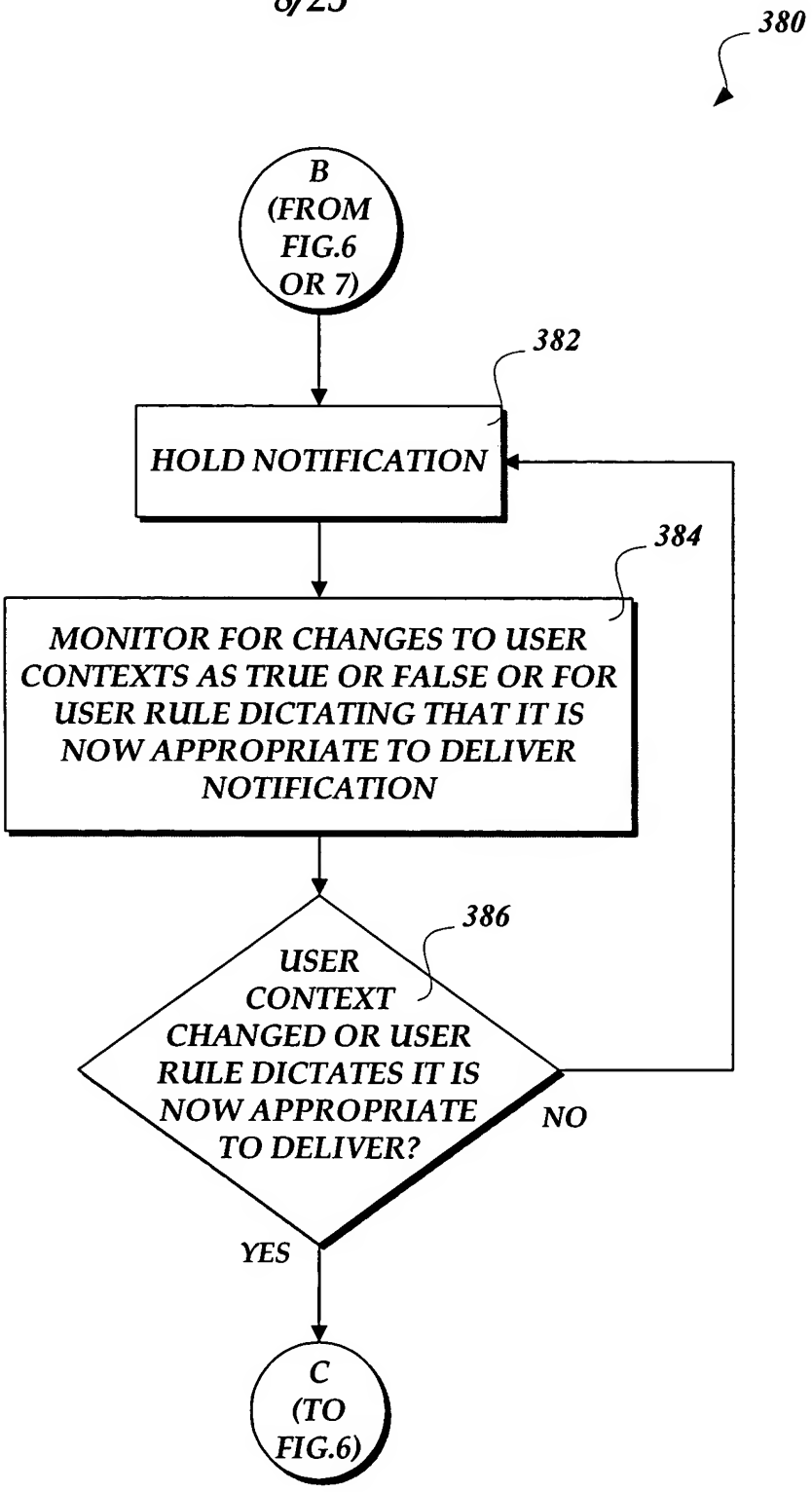
380
B (FROM FIG.6 OR 7)
382
HOLD NOTIFICATION
384
MONITOR FOR CHANGES TO USER CONTEXTS AS TRUE OR FALSE OR FOR USER RULE DICTATING THAT IT IS NOW APPROPRIATE TO DELIVER NOTIFICATION
386
USER CONTEXT CHANGED OR USER RULE DICTATES IT IS NOW APPROPRIATE TO DELIVER?
NO
YES
C (TO FIG.6)

Fig.8.

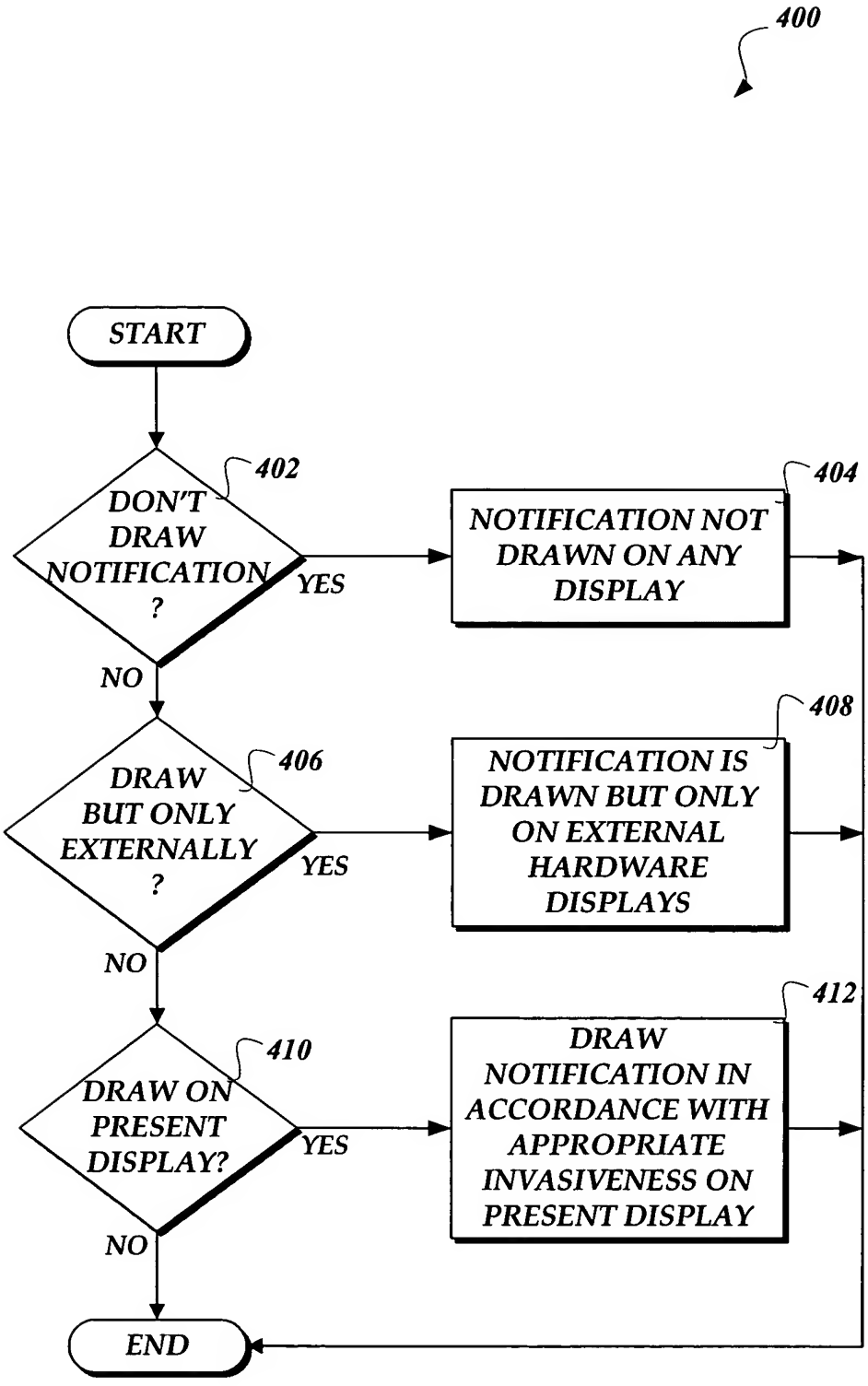
400
START
402
DON'T DRAW NOTIFICATION ?
YES
404
NOTIFICATION NOT DRAWN ON ANY DISPLAY
NO
406
DRAW BUT ONLY EXTERNALLY ?
YES
408
NOTIFICATION IS DRAWN BUT ONLY ON EXTERNAL HARDWARE DISPLAYS
NO
410
DRAW ON PRESENT DISPLAY?
YES
412
DRAW NOTIFICATION IN ACCORDANCE WITH APPROPRIATE INVASIVENESS ON PRESENT DISPLAY
NO
END

Fig.9.

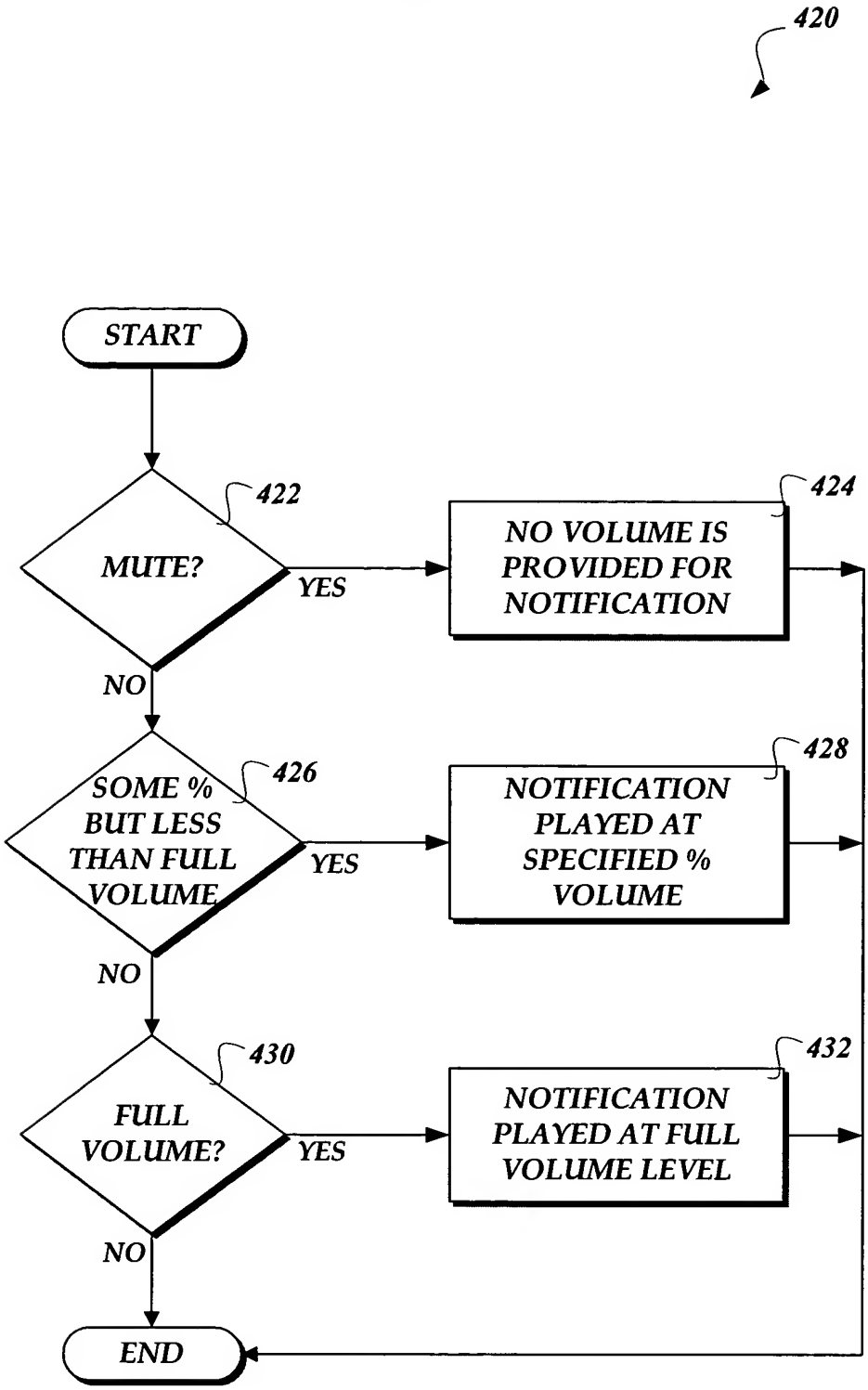
420
START
422
MUTE?
YES
424
NO VOLUME IS PROVIDED FOR NOTIFICATION
NO
426
SOME % BUT LESS THAN FULL VOLUME
YES
428
NOTIFICATION PLAYED AT SPECIFIED % VOLUME
NO
430
FULL VOLUME?
YES
432
NOTIFICATION PLAYED AT FULL VOLUME LEVEL
NO
END

Fig.10.

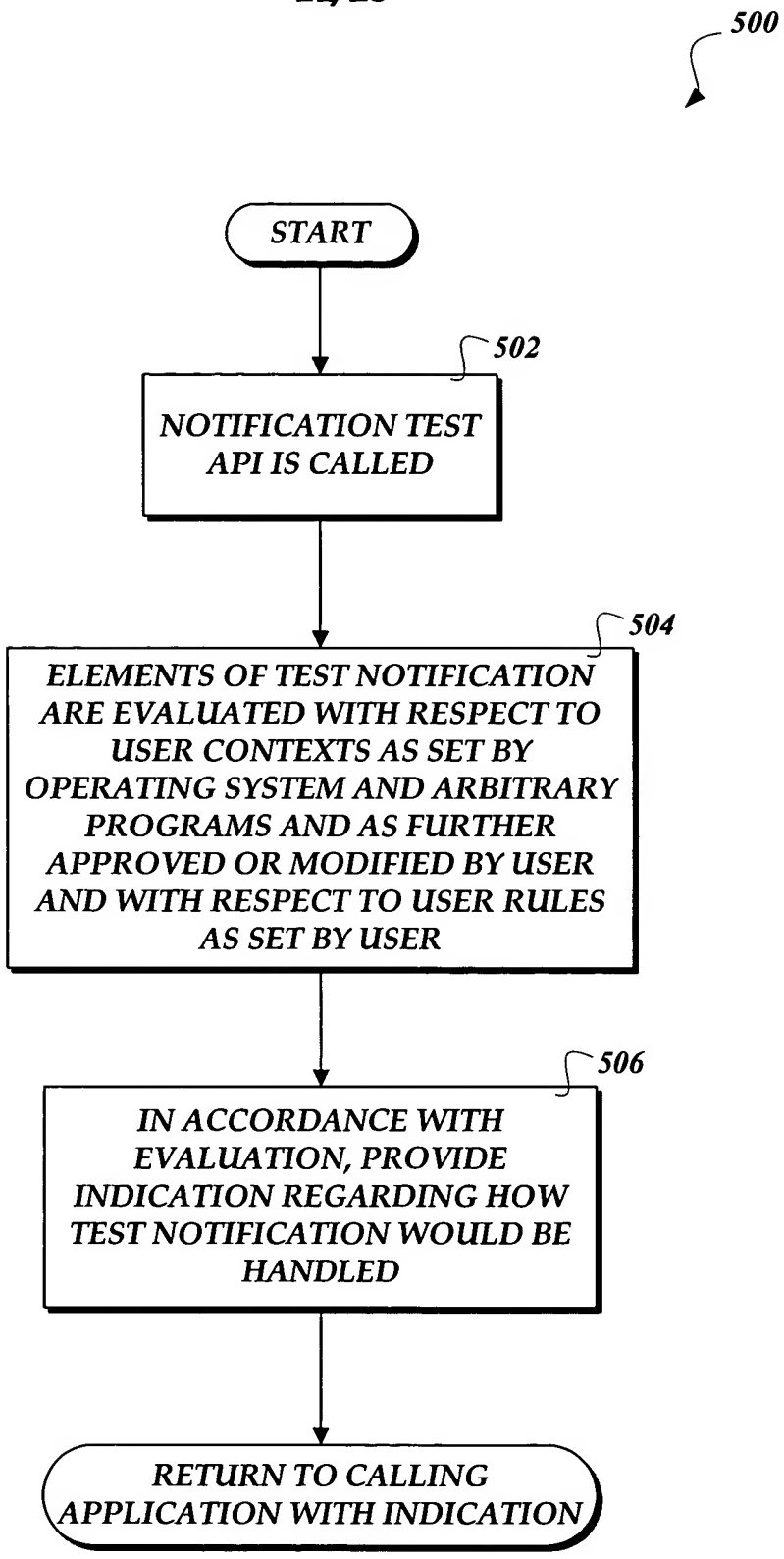
500
START
502
NOTIFICATION TEST API IS CALLED
504
ELEMENTS OF TEST NOTIFICATION ARE EVALUATED WITH RESPECT TO USER CONTEXTS AS SET BY OPERATING SYSTEM AND ARBITRARY PROGRAMS AND AS FURTHER APPROVED OR MODIFIED BY USER AND WITH RESPECT TO USER RULES AS SET BY USER
506
IN ACCORDANCE WITH EVALUATION, PROVIDE INDICATION REGARDING HOW TEST NOTIFICATION WOULD BE HANDLED
RETURN TO CALLING APPLICATION WITH INDICATION

Fig.11.

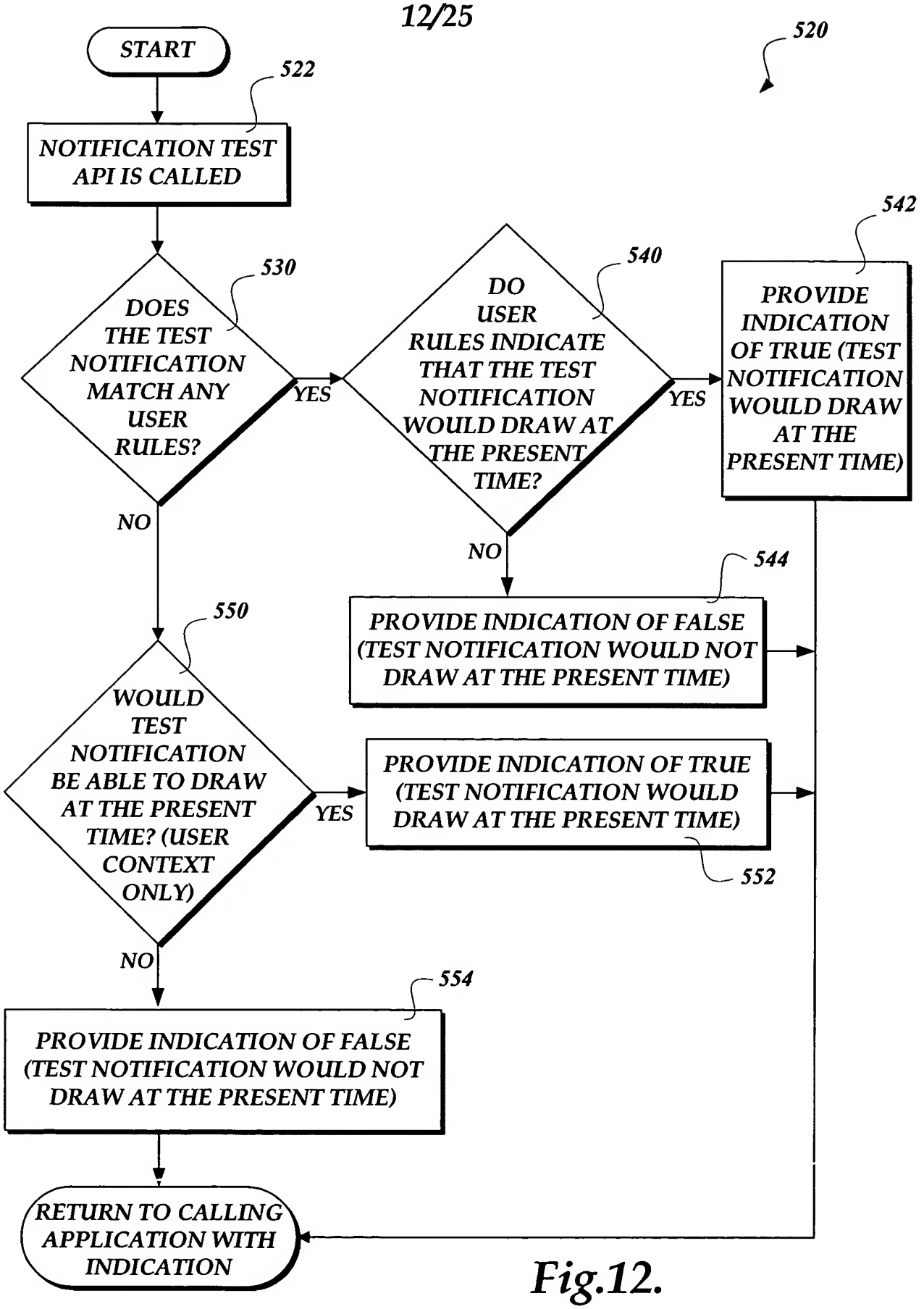
520
START
522
NOTIFICATION TEST API IS CALLED
530
DOES THE TEST NOTIFICATION MATCH ANY USER RULES?
YES
NO
540
DO USER RULES INDICATE THAT THE TEST NOTIFICATION WOULD DRAW AT THE PRESENT TIME?
YES
NO
542
PROVIDE INDICATION OF TRUE (TEST NOTIFICATION WOULD DRAW AT THE PRESENT TIME)
544
PROVIDE INDICATION OF FALSE (TEST NOTIFICATION WOULD NOT DRAW AT THE PRESENT TIME)
550
WOULD TEST NOTIFICATION BE ABLE TO DRAW AT THE PRESENT TIME? (USER CONTEXT ONLY)
YES
NO
PROVIDE INDICATION OF TRUE (TEST NOTIFICATION WOULD DRAW AT THE PRESENT TIME)
552
554
PROVIDE INDICATION OF FALSE (TEST NOTIFICATION WOULD NOT DRAW AT THE PRESENT TIME)
RETURN TO CALLING APPLICATION WITH INDICATION

Fig.12.

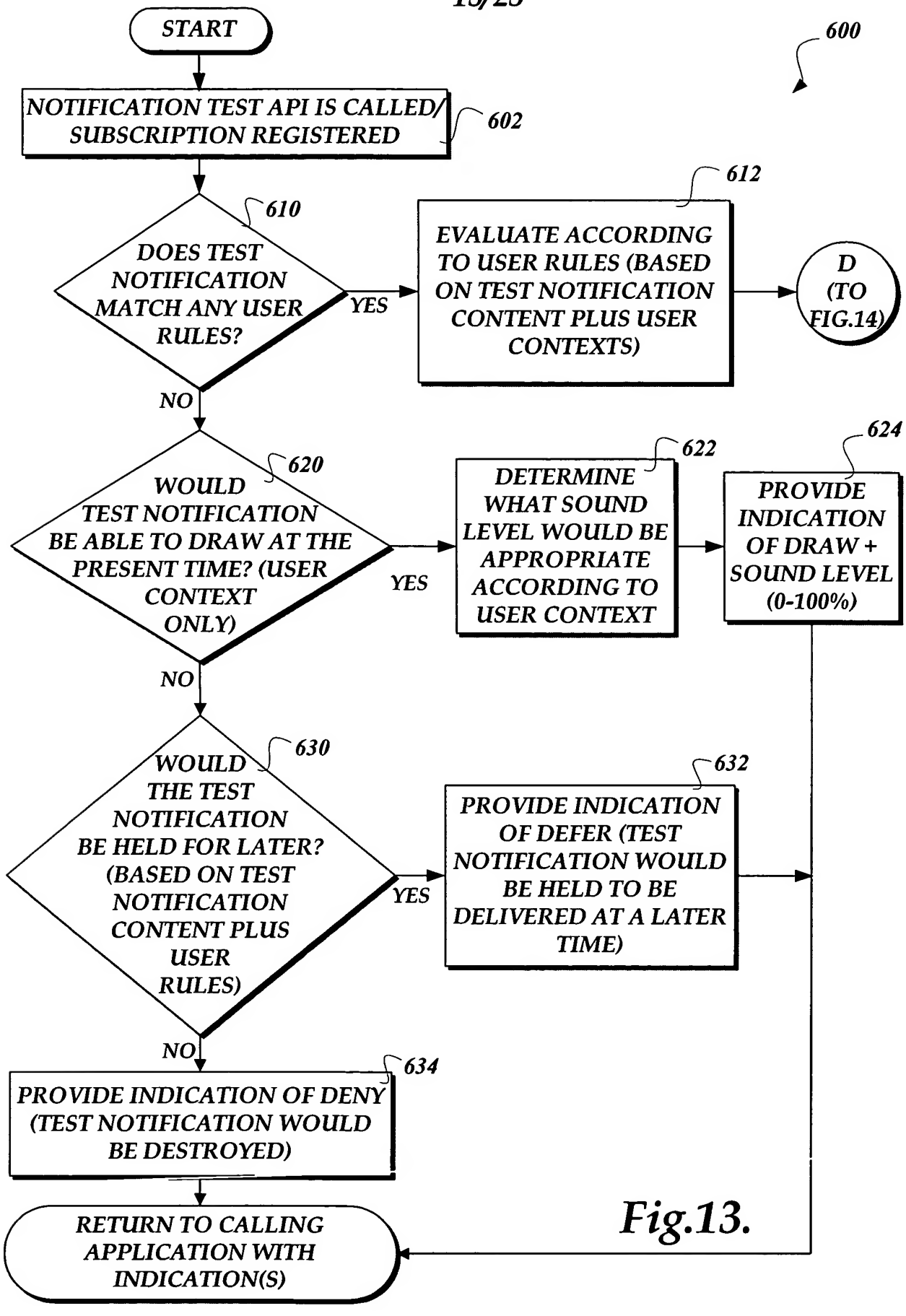
START
600
NOTIFICATION TEST API IS CALLED/ SUBSCRIPTION REGISTERED
602
610
DOES TEST NOTIFICATION MATCH ANY USER RULES?
YES
612
EVALUATE ACCORDING TO USER RULES (BASED ON TEST NOTIFICATION CONTENT PLUS USER CONTEXTS)
D (TO FIG.14)
NO
620
WOULD TEST NOTIFICATION BE ABLE TO DRAW AT THE PRESENT TIME? (USER CONTEXT ONLY)
YES
622
DETERMINE WHAT SOUND LEVEL WOULD BE APPROPRIATE ACCORDING TO USER CONTEXT
624
PROVIDE INDICATION OF DRAW + SOUND LEVEL (0-100%)
NO
630
WOULD THE TEST NOTIFICATION BE HELD FOR LATER? (BASED ON TEST NOTIFICATION CONTENT PLUS USER RULES)
YES
632
PROVIDE INDICATION OF DEFER (TEST NOTIFICATION WOULD BE HELD TO BE DELIVERED AT A LATER TIME)
NO
634
PROVIDE INDICATION OF DENY (TEST NOTIFICATION WOULD BE DESTROYED)
RETURN TO CALLING APPLICATION WITH INDICATION(S)

Fig.13.

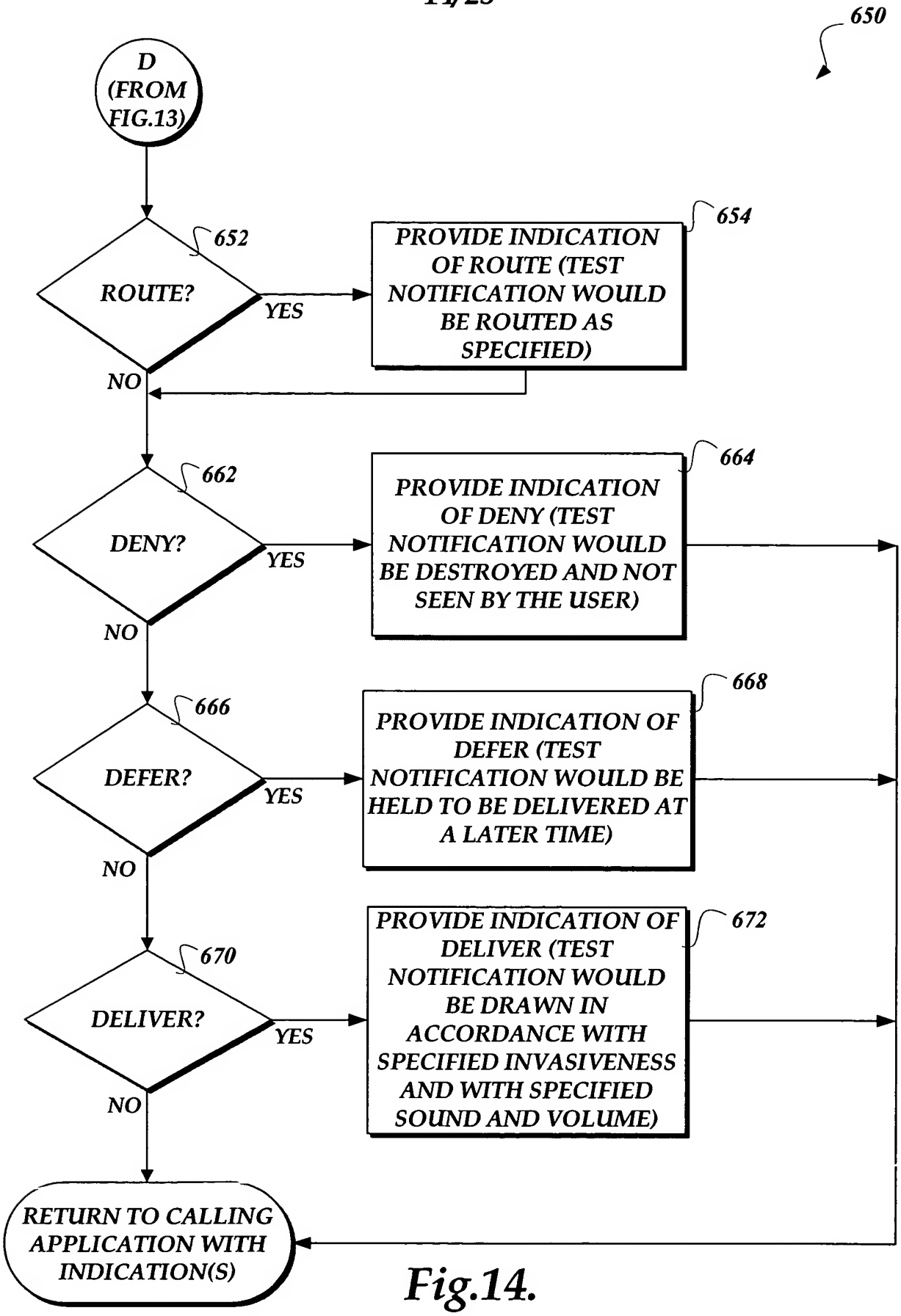
650
D (FROM FIG.13)
652
ROUTE?
YES
NO
654
PROVIDE INDICATION OF ROUTE (TEST NOTIFICATION WOULD BE ROUTED AS SPECIFIED)
662
DENY?
YES
NO
664
PROVIDE INDICATION OF DENY (TEST NOTIFICATION WOULD BE DESTROYED AND NOT SEEN BY THE USER)
666
DEFER?
YES
NO
668
PROVIDE INDICATION OF DEFER (TEST NOTIFICATION WOULD BE HELD TO BE DELIVERED AT A LATER TIME)
670
DELIVER?
YES
NO
672
PROVIDE INDICATION OF DELIVER (TEST NOTIFICATION WOULD BE DRAWN IN ACCORDANCE WITH SPECIFIED INVASIVENESS AND WITH SPECIFIED SOUND AND VOLUME)
RETURN TO CALLING APPLICATION WITH INDICATION(S)

Fig.14.

1500A

```
/// <ExternalAPI/>                                                    1510

        /// <summary>
         /// Use this object to send a balloon Notification to the user. You can derive
this class if you need to add custom properties to this object so that you can have more
context when a NotificationClickedEvent arrives.
        /// </summary>
        public class Notification
        {
        public:
          Notification();                                                  1520

          /// <summary>
            /// This is the icon for your notification. Minimum size is 16x16, and
maximum size is 80x80.
          /// </summary>
          property ImageData HeaderIcon; get, set                          1530

          /// <summary>
          /// This is the title text for you notification.
          /// </summary>                                                   1540
          property String HeaderText; get, set

          /// <summary>
          /// This is the main body text of your notification.
          /// </summary>                                                   1550
          property String BodyText; get, set

          /// <summary>
           /// Set this property to true if you want the user to be able to click inside a
notification to trigger some event.
          /// </summary>
          property bool IsBodyClickable; get, set
          void AddButton(NotificationButton button);
          event NotificationClickedEventHandler
NotificationClickedEvent;
```

Fig. 15A.

1500B

1560

```
        /// <summary>
        /// Call this method after setting all the relevant Notification properties
when you are ready to send this so that it will be shown to the user. This method
returns immediately.
        /// </summary>
        void Send();
```

1570

```
        /// <summary>
        /// Use this to update a notification that has already been sent. You can
change some properties and then call Update() to have the changes be reflected.
        /// </summary>
        void Update();
```

1580

```
        /// <summary>
        /// Call this if you want to know if a particular notification would actually
draw on screen at this time.
        /// </summary>
        bool WouldNotificationShow();
```

1590

```
        /// <summary>
        /// Call this when you want to recall a notification that was already sent,
but that has become obsolete.
        /// </summary>
        void Revoke();
    };
```

Fig. 15B.

1600

```
/// <ExternalAPI/>                                          1610

/// <summary>
 /// This class helps you manage your context changes. Your custom contexts
are defined in your application manifest. This class lets you set your context to true, or
reset it to false. An example of context is the FullScreen context (that context is
managed by the operating system) which is true when any application is full screen.
The default value for any context is always false.
/// </summary>
public class NotificationContext
{
public:                                                     1620

  /// <summary>
    /// Construct a context object for a particular context, identified by its
Guid.
  /// </summary>
  NotificationContext(Guid ContextGuid);                    1630

  /// <summary>
   /// Change the value of the context. All contexts for an application will be
reset to false when the application terminates or dies.
  /// </summary>
  property bool Context; set only
};
```

Fig. 16.

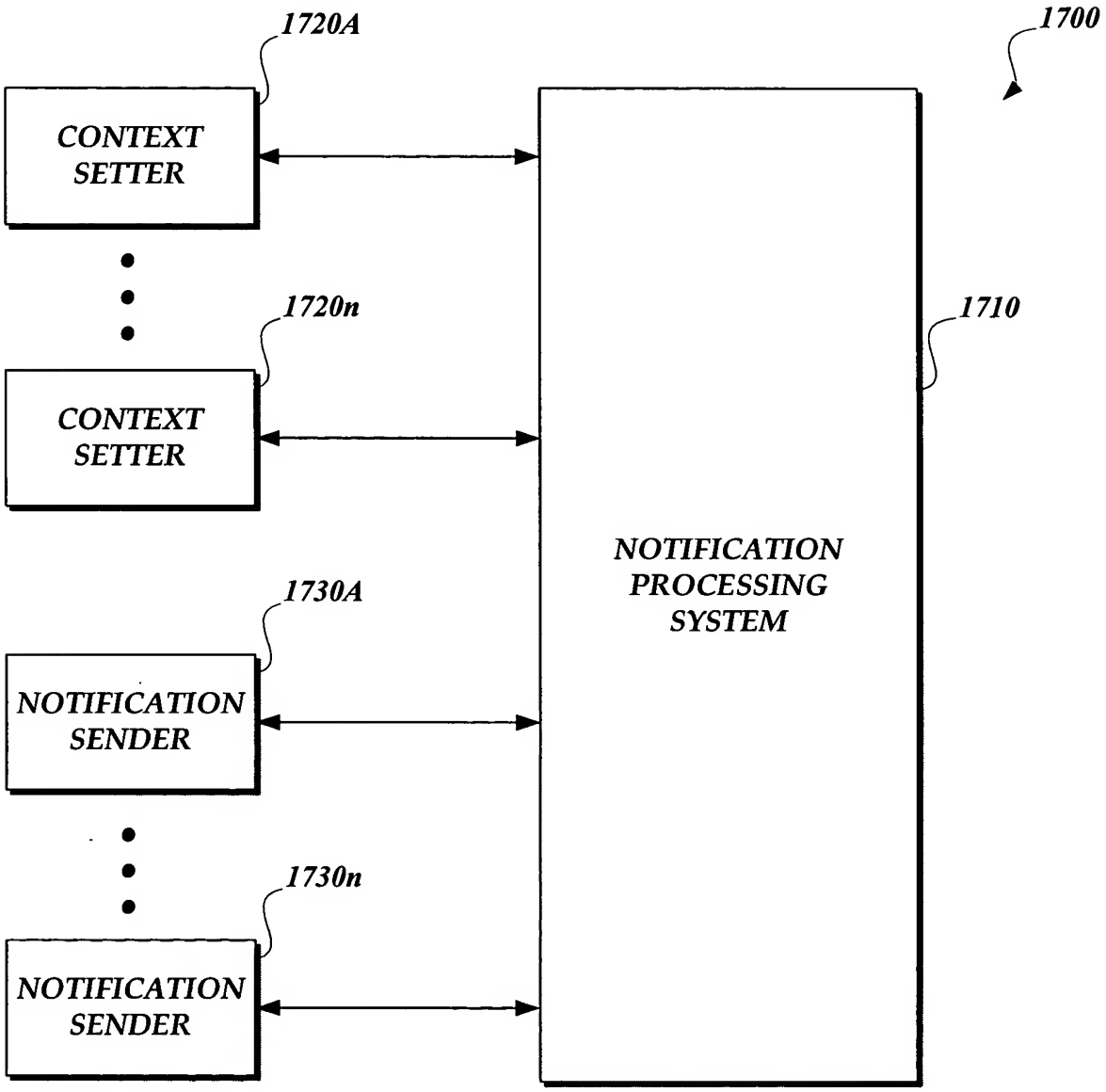
1720A
1700
CONTEXT SETTER
1720n
1710
CONTEXT SETTER
NOTIFICATION PROCESSING SYSTEM
1730A
NOTIFICATION SENDER
1730n
NOTIFICATION SENDER

Fig. 17.

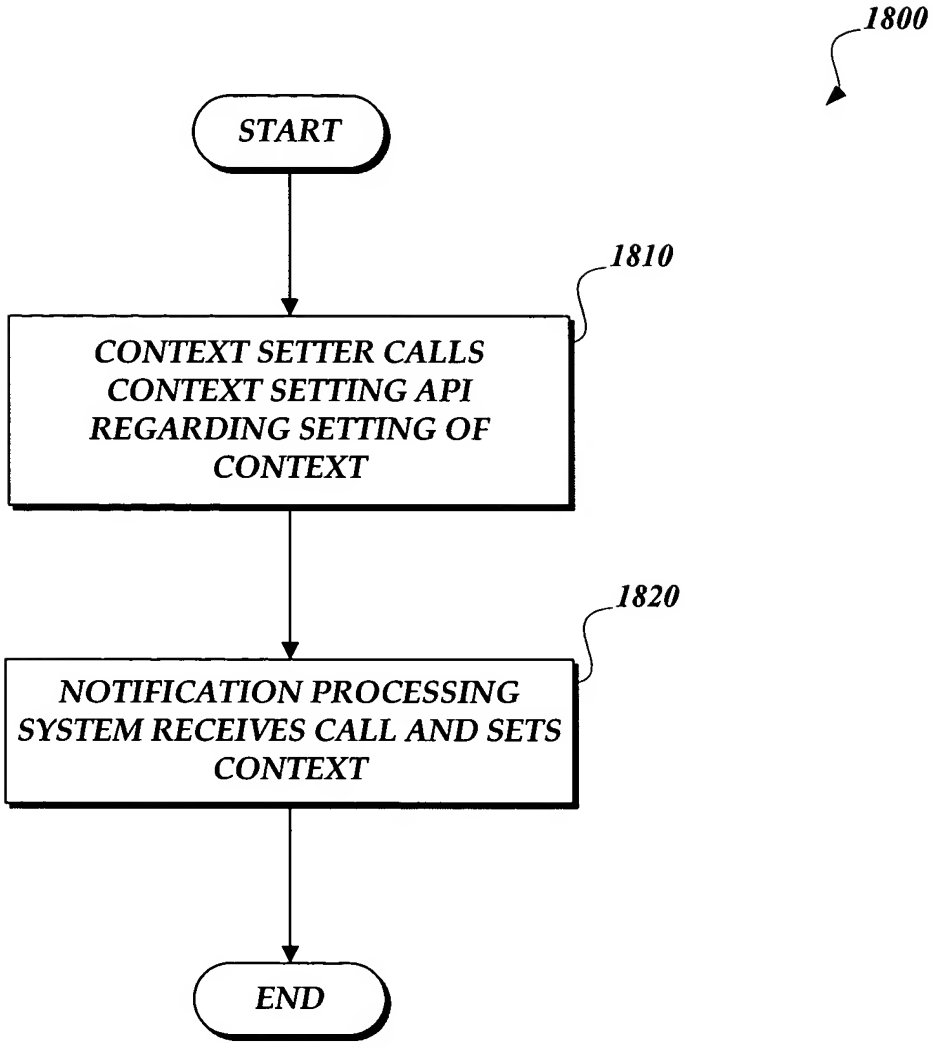
1800
START
1810
CONTEXT SETTER CALLS CONTEXT SETTING API REGARDING SETTING OF CONTEXT
1820
NOTIFICATION PROCESSING SYSTEM RECEIVES CALL AND SETS CONTEXT
END

Fig. 18.

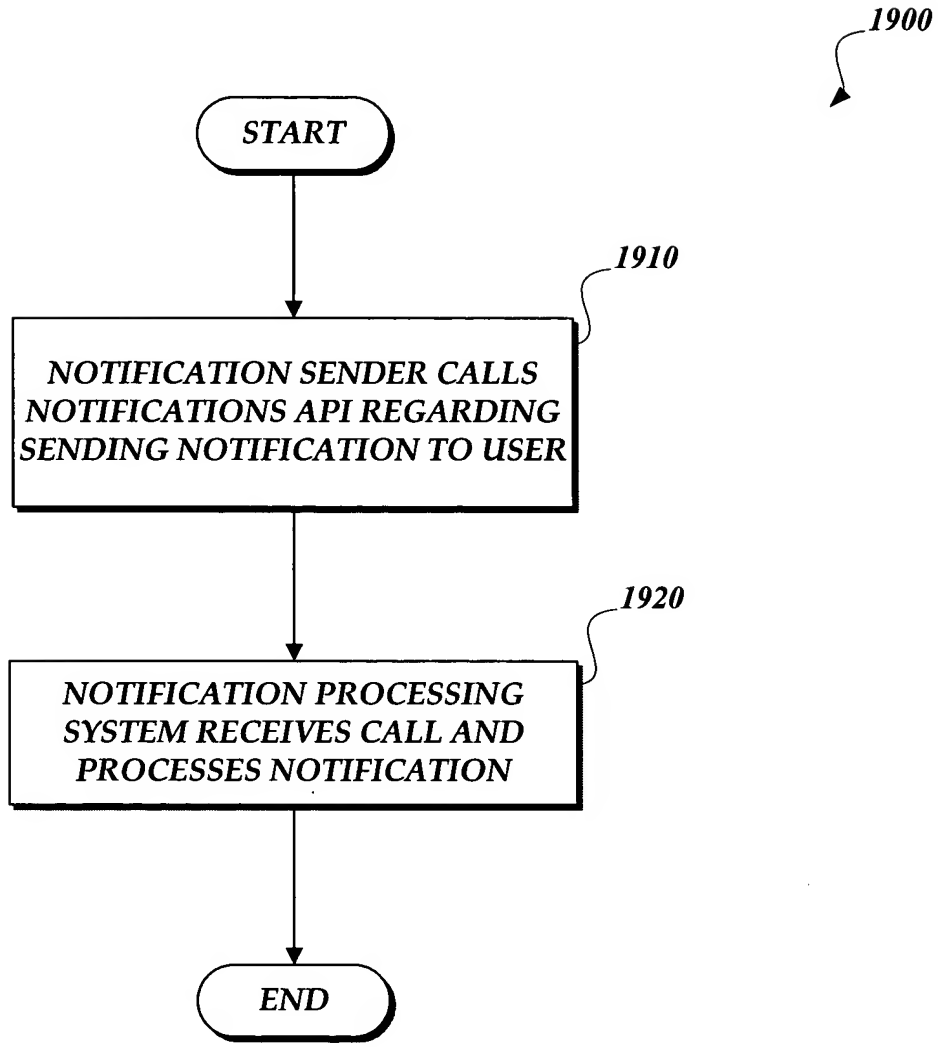
1900
START
1910
NOTIFICATION SENDER CALLS NOTIFICATIONS API REGARDING SENDING NOTIFICATION TO USER
1920
NOTIFICATION PROCESSING SYSTEM RECEIVES CALL AND PROCESSES NOTIFICATION
END

Fig. 19.

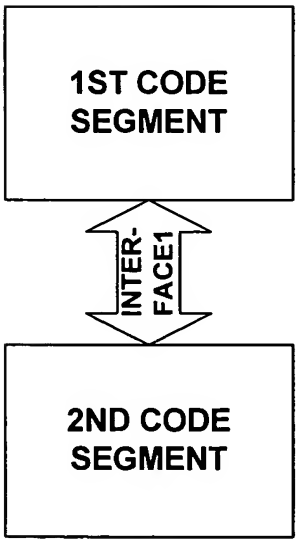

Fig. 20A

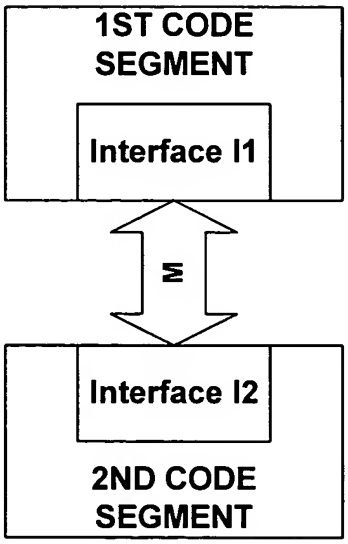

Fig. 20B

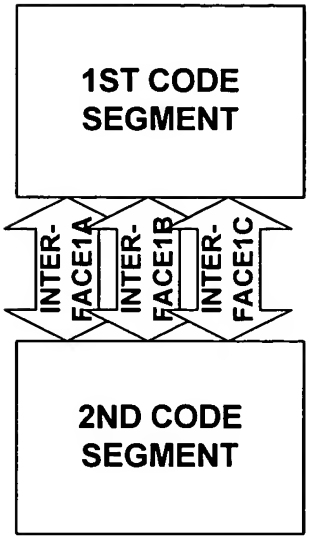

Fig. 20C

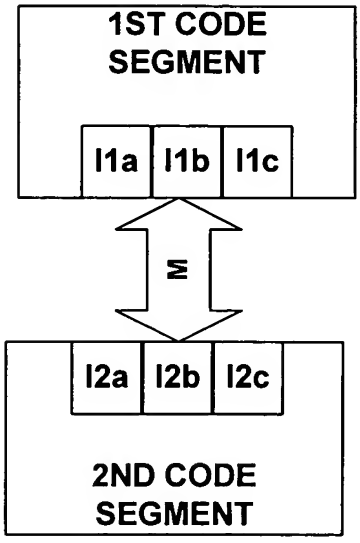

Fig. 20D

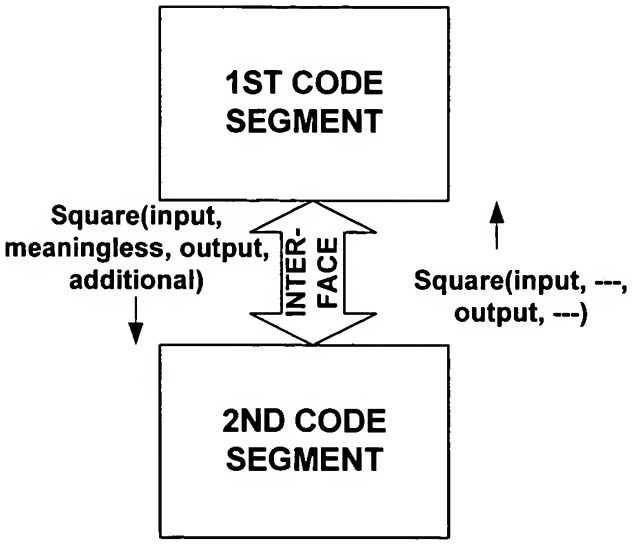

Fig. 20E

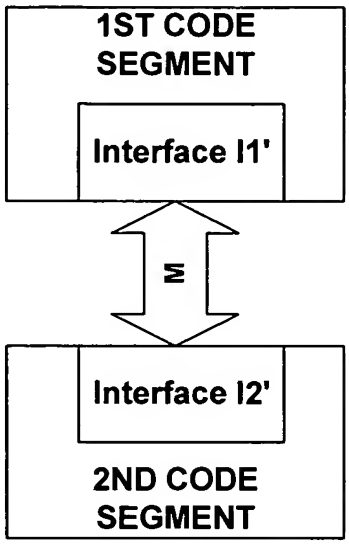

Fig. 20F

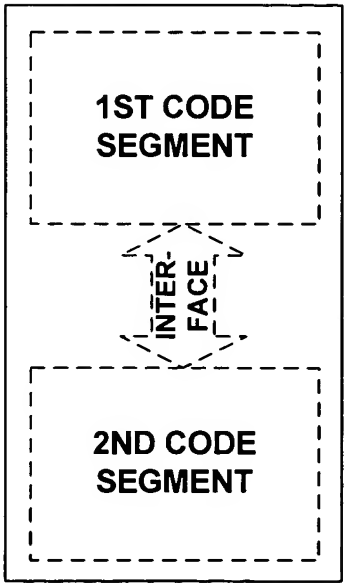

Fig. 20G

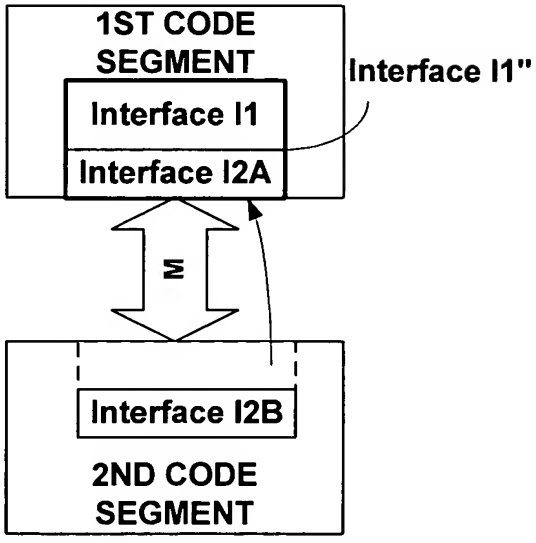

Fig. 20H

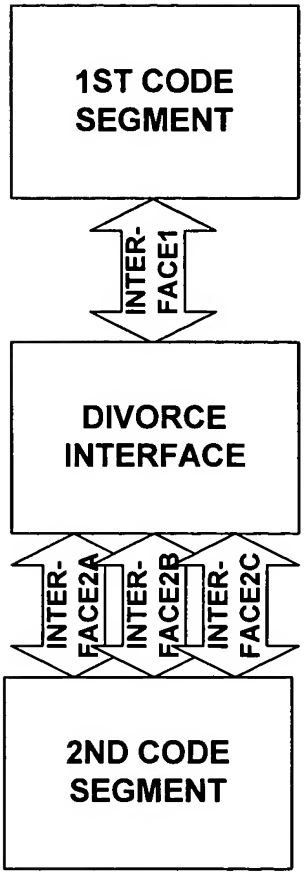

Fig. 20I

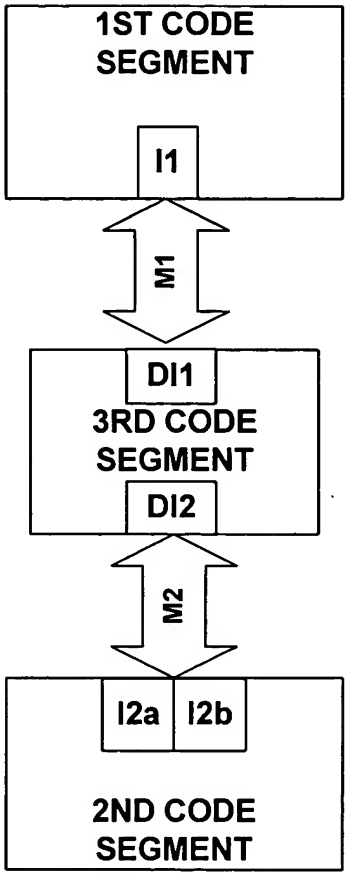

Fig. 20J

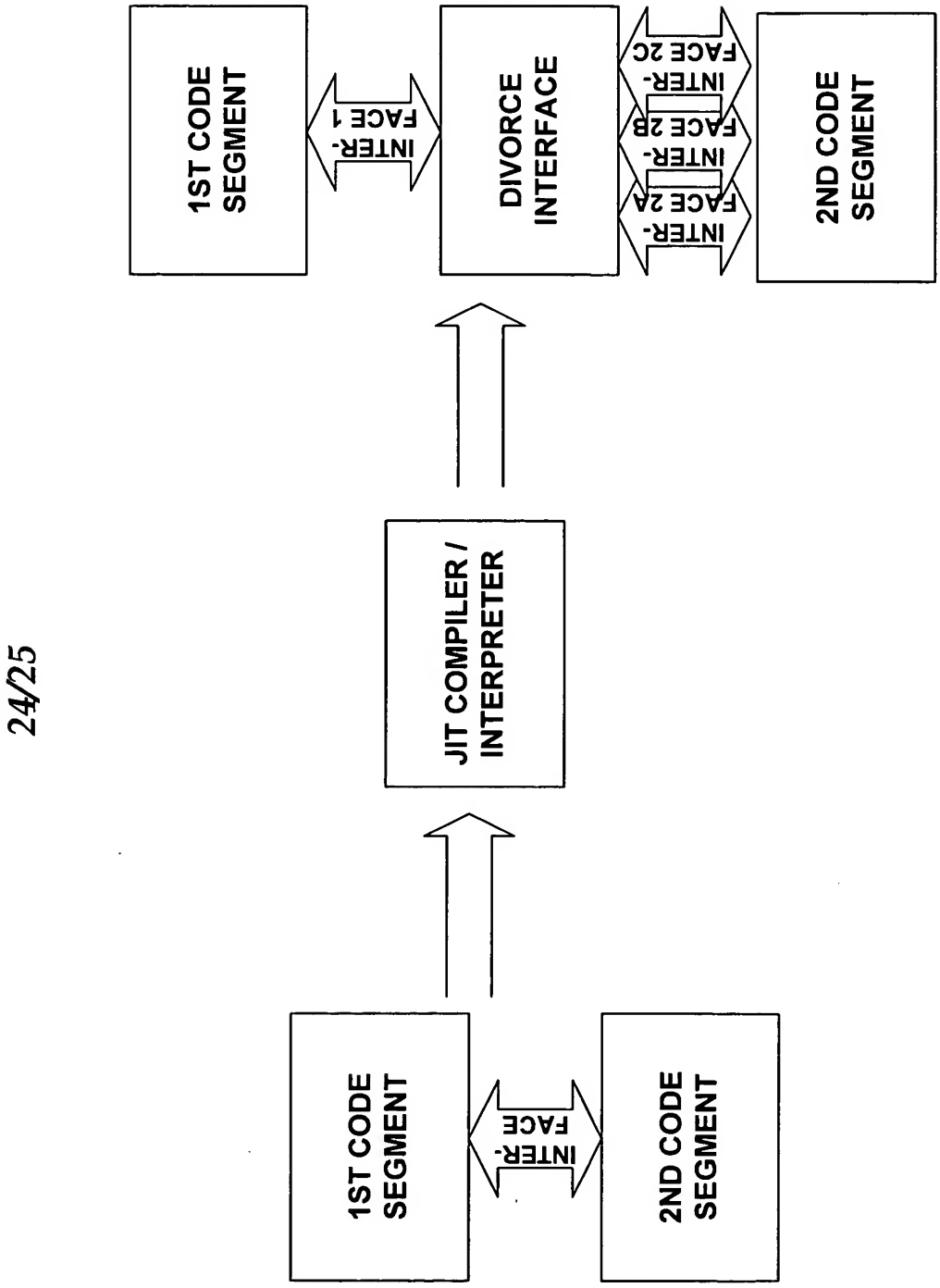
1ST CODE SEGMENT
INTER-FACE 1
DIVORCE INTERFACE
INTER-FACE 2C
INTER-FACE 2B
INTER-FACE 2A
2ND CODE SEGMENT
JIT COMPILER / INTERPRETER
1ST CODE SEGMENT
INTER-FACE
2ND CODE SEGMENT

Fig. 20K

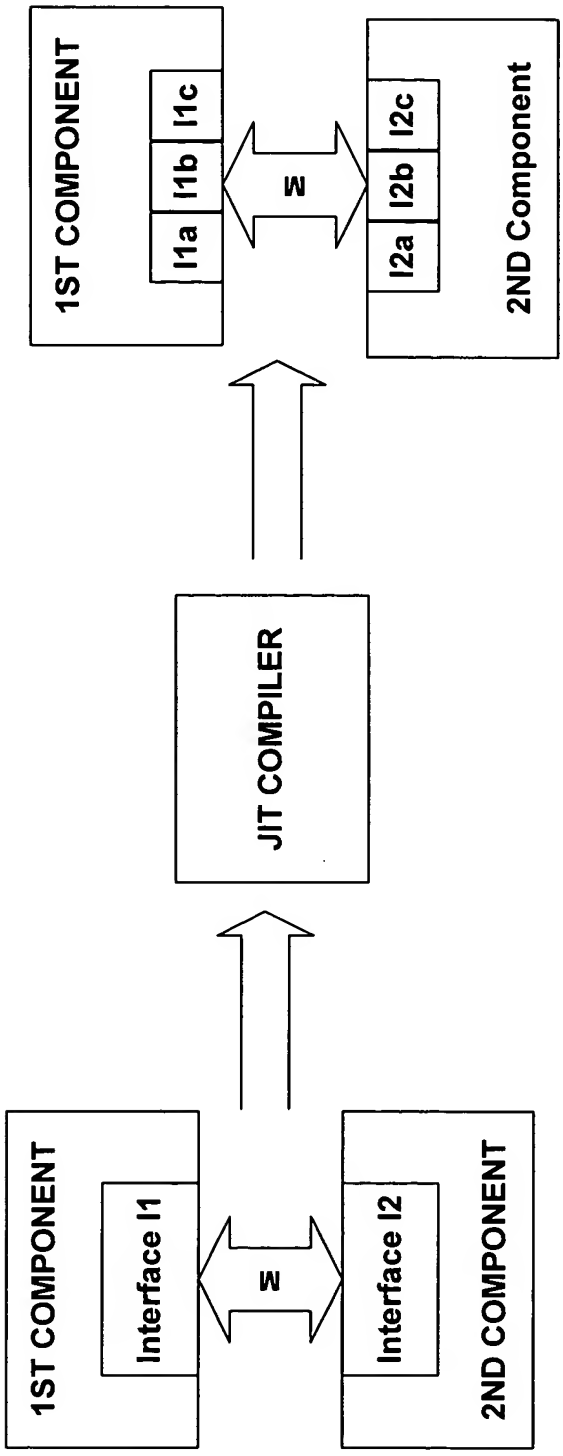
1ST COMPONENT
I1a
I1b
I1c
2ND Component
I2a
I2b
I2c
JIT COMPILER
1ST COMPONENT
Interface I1
Interface I2
2ND COMPONENT

Fig. 20L